REMARKS

The Examiner is thanked for the performance of a thorough search.

By this amendment, no claims have been added, cancelled, or amended. Consequently, Claims 5-14, 16-18, 23-32, and 34-36 are currently pending.

The Examiner is thanked for the allowance of Claims 16-18 and 34-36.

The first paragraph of the Office Action incorrectly stated that Claim 23 was cancelled, and that Claims 5-14, 16-18, and 23-36 were presented for examination. This statement is in error, as Claims 5-14, 16-18, 23-32, and 34-36 are currently pending, and are presented for examination.

Filed Ids Has Not Been Acknowledged

The Examiner is thanked for the transmittal of an initialed form PTO-1449 acknowledging receipt and consideration of the information disclosure statements (individually an "IDS") filed on January 28, 2005, January 4, 2005, December 2, 2004, September 23, 2004, July 19, 2004, May 18, 2004, November 17, 2003, October 7, 2003, October 20, 2002, and May 4, 2002. However, the Applicants have not received an initialed form PTO-1449 acknowledging receipt and consideration of the IDS filed on April 17, 2003 (the "April 17, 2003 IDS"). Thus, the Applicants respectfully request receipt of an initialed form PTO-1449 acknowledging receipt and consideration of the April 17, 2003 IDS.

Summary of the Rejections

The Office Action contains a request for the Applicants to submit formal drawings.

Claims 12-14 and 30-32 have been rejected under 35 U.S.C. § 112, second paragraph, as allegedly being indefinite.

Claims 5-7, 9, 23-25 and 27 have been rejected under 35 U.S.C. § 103(a) as allegedly unpatentable over IBM Technical Disclosure Bulletin entitled "Memory Queue Priority Mechanism for a RISC Processor by Bull et al. ("*Bull*") in view of U.S. Patent No. 5,966,706 issued to Biliris et al. ("*Biliris*").

Claims 8 and 26 have been rejected under 35 U.S.C. § 103(a) as allegedly unpatentable over Bull in view of U.S. Patent No. 5,933,838 issued to Lomet ("*Lomet*").

Claims 12-14 and 30-32 have been rejected under 35 U.S.C. § 102(b) as allegedly unpatentable over an article entitled "Recovering in Large Distributed Systems with Replicated Data" by Peter Triantafillou ("*Peter*").

Claims 10-11 and 28-29 have been rejected under 35 U.S.C. § 102(e) as allegedly unpatentable over *Biliris*.

The rejections are respectfully traversed.

Formal Drawings Have Already Been Submitted

While the Office Action contains a request for the Applicants to submit formal drawings, the Office Action contains no explanation as to why the formal drawings previously submitted are deficient. In a phone call placed by the Applicant's attorney, Chris Brokaw, to the Examiner, on June 2, 2005, the Examiner explained that the request to submit formal drawings, contained in the Office Action, was a general request that was made without specific reference to the figures of the present application. Consequently, the Applicants respectfully submit that the formal figures

previously submitted render the request for formal figures moot.

Claims 12-14 and 30-32 are Definite Under 35 U.S.C. § 112, Second Paragraph

Claims 12-14 and 30-32 have been rejected under 35 U.S.C. § 112, second paragraph, on grounds that the feature of "if it is unclear whether a particular version of a data item has been written to disk" (hereinafter referred to as the "identified language") is allegedly vague and indefinite. The Office Action argues "it is not clear as to how one having ordinary skill in the art would determine whether a particular version of a data item has been written to disk."

The Applicants respectfully submit that the plain meaning and scope of the identified language is clear and precise on its face.

Further, the identified language, when viewed with the benefit of the Applicants' specification, is clear both in meaning and in scope. For example, paragraphs 132-135 of the Applicants' specification teaches:

> Under certain circumstances, it can be unclear whether a write-to-disk operation is successfully performed. For example, if a node writing a data item to disk fails during the write operation, it may be unclear whether the failure occurred before or after the write operation was successfully completed. Similarly, if the node on which the master of a particular data item resides fails, the failure may result in a loss of information about the data item. Such information may include information that indicates the last version of the data item to be written to disk.
>
> When a situation occurs where it is unclear whether a write-to-disk operation was successfully performed, the issue may be resolved by scanning the data items on disk to determine their versions. However, scanning the disk as part of the recovery operation would consume a significant amount of time and resources, and may unduly delay the availability of the data.
>
> According to one aspect of the invention, the need to scan the on-disk data items is avoided by (1) if it is unclear whether a particular version of a data item has been written to disk and the recovery information (e.g. redo log) indicates that the particular version was written to disk, causing the recovery process to assume that the particular data item was successfully written to disk, and (2) marking all earlier cached versions of that data item as "suspect". After the recovery operation, the system may then proceed under the opposite assumption. Specifically, the system proceeds under the assumption that the particular version of the data item was not

written to disk. However, prior to writing any suspect version of the data item to disk, the system reads the version of the data item that resides on disk. If the on-disk version of the data item is more recent, then the write-to-disk operation is not performed, and the master is informed of which version is on disk. Optionally, the master then sends write-notification messages to all nodes that hold versions that are covered by the version that is on the disk. On the other hand, the data item is recovered.

Similarly, when a node requests the current version of a data item, the requesting node cannot be supplied a suspect version of the data item because the disk may contain a more recent version of the data item. Instead, the on-disk version of the data item is read from disk. If the version of the data item that is read from disk is the most recent version, then that version is provided to the requesting node. If the on-disk version of the data item is not the most recent version, then the most recent version is created based on the recovery information maintained in the redo log of the node that had failed.

The identified language, when read in conjunction with the Applicants' specification, particularly points out and distinctly claims the subject matter which the Applicants' regard as their invention in conformance with 35 U.S.C. § 112, second paragraph.

The Office Action objects to the identified language because, allegedly, it is not clear as to how one having ordinary skill in the art would determine whether a particular version of a data item has been written to disk. First, the portion of the Applicants' specification cited above clearly provides a teaching of (a) situations where it may be unclear whether a particular version of a data item has been written to disk and (b) mechanisms for determining whether a particular version of a data item has been written to disk. Thus, the concerns of the Office Action are directly addressed by the teaching of the above portion of the Applicants' specification.

Second, the issue of how a person of ordinary skill in the art would determine whether a particular version of a data item has been written to disk is not relevant to the question whether the identified language is vague and indefinite. In other words, determining how one would potentially answer a question is not determinative of whether the question itself is clear in meaning. In the present case, the scope of coverage being sought by the Applicants, in Claims 12-

14 and 30-32 is clear to both those of ordinary skill in the art and to those with the benefit of the teachings of the Applicants' specification.

Consequently, for at least the above reasons, it is respectfully submitted that the identified language in Claims 12-14 and 30-32 conforms to U.S.C. § 112, second paragraph. As a result, the rejection to Claims 12-14 and 30-32, based on U.S.C. § 112, second paragraph, is respectfully requested to be removed.

The Pending Claims Are Patentable Over The Cited Art

Each of the pending claims recites a combination of features that are not disclosed, taught, or suggested by the cited art, either individually or in combination. Each pending claim shall be discuss individually below.

Claim 5

Claim 5 recites the features of:

> A method of managing information about where to begin recovery after a failure, the method comprising the steps of:
> in a particular node of a multiple-node system, maintaining both
> a single-failure queue that indicates where within a recovery log to begin recovery after a failure of said node, and
> a multiple-failure queue that indicates where within said recovery log to begin recovery after a failure of said node and one or more other nodes in said multiple-node system;
> in response to a dirty data item being written to persistent storage, removing an entry for said data item from both said single-failure queue and said multiple-failure queue; and
> in response to a dirty data item being sent to another node of said multiple-node system without first being written to persistent storage, removing an entry for said data item from said single-failure queue without removing the entry for said data item from said multiple-failure queue.

The above combination of elements is not disclosed, taught, or suggested by *Bull* and *Biliris*, either individually or in combination.

Bull is directed towards a priority scheme used in a RISC microprocessor memory queue. The priority scheme of *Bull* improves processor performance and avoids bus deadlock situations in coherent memory systems (see page 319).

Significantly, there are substantial, fundamental differences between the approach of *Bull* and that taken by the pending claim. *Bull* is directed towards the memory queue of a microprocessor. As a result, *Bull* contains no suggestion of persistently storing any data items to disk. Further, *Bull* is directed towards a priority scheme for use in a single microprocessor. Consequently, the approach of *Bull* can only be used within a single node. As a result, *Bull* lacks any suggestion of a multiple-node system.

In view of the substantial, fundamental differences between the approach of *Bull* and that taken by the pending claims, *Bull* does not show any element of Claim 5. The Office Action acknowledges, "*Bull* does not explicitly disclose a checkpoint and recovery algorithm involving single failure queues and multiple failure queues." While Claim 5 does not contain any elements that make reference to an algorithm, it is clear from the position of the Office Action that the Office Action is acknowledging that *Bull* does not disclose, teach, or suggest the elements of "in a particular node of a multiple-node system, maintaining both a single-failure queue that indicates where within a recovery log to begin recovery after a failure of said node, and a multiple-failure queue that indicates where within said recovery log to begin recovery after a failure of said node and one or more other nodes in said multiple-node system."

Instead, *Bull* is relied upon to show the element of "in response to a dirty data item being written to persistent storage, removing an entry for said data item from both said single-failure queue and said multiple-failure queue" featured in Claim 5. Since the Office Action acknowledges that *Bull* fails to teach a single-failure queue and a multiple failure queue as claimed, *Bull* cannot possibly suggest this element, because this element requires that an entry for

the data item be removed from both the single-failure queue and the multiple-failure queue.

Additionally, *Bull* lacks any teaching of writing any data items to a persistent storage. For example, the portion cited by the Office Action to show this element (page 320, "snoop copyback and program initiated") merely describes writing data from the cache (which is volatile memory) of a microprocessor to the bus memory, which is also volatile memory. In other words, a bus memory is not a persistent storage. Consequently, this element cannot be disclosed, taught, or suggested by *Bull*.

Similarly, *Bull* fails to suggest the element of "in response to a dirty data item being sent to another node of said multiple-node system without first being written to persistent storage, removing an entry for said data item from said single-failure queue without removing the entry for said data item from said multiple-failure queue" featured in Claim 5. As explained above, since the Office Action acknowledges that *Bull* fails to teach a single-failure queue and a multiple failure queue as claimed, *Bull* cannot possibly suggest this element, because this element requires that an entry for the data item be removed from the single-failure queue without removing the entry for the data item from the multiple-failure queue.

Additionally, *Bull* lacks any teaching of sending data items to other nodes of a multiple-node system. The portion of *Bull* cited to show this element (page 320, lines 25-27) merely describes transferring a data item from the memory queue of a microprocessor to the memory bus of the same microprocessor. Hence, the data item is never sent to another node, as it remains within the same node, e.g., in this case, a single microprocessor. Consequently, this element cannot be disclosed, taught, or suggested by *Bull*.

As one or more elements of Claim 5 are not disclosed, taught, or suggested by the cited art, it is respectfully submitted that Claim 5 is patentable over the cited art, and is in condition for allowance.

Claim 10

Claim 10 recites the features of:

> A method for recovering after a failure, the method comprising the steps of:
> determining whether the failure involves only one node; and
> if the failure involves only said one node, then performing recovery by applying a recovery log of said node beginning at a first point in the recovery log; and
> if the failure involves one or more nodes in addition to said one node, then performing recovery by applying said recovery log of said node beginning at a second point in the recovery log;
> wherein said first point is different from said second point.

The above combination of elements is not disclosed, taught, or suggested by *Biliris*.

Biliris is directed towards performing local logging in a distributed database management computer system. When a first node updates a first database page, the first node generates a log record. The first node determines whether it manages the first database page. If so, then the first node writes the log record to a log storage local to the first node. If not, then the first node determines whether the first node includes a local log storage. If the first node includes a local log storage, then the first node writes the log record to the local log storage, even though the first node does not manage the first database page (see Abstract).

Significantly, because each node performs local logging in the approach of *Biliris*, it is necessary for a crashed node to not only scan its own log file, but to communicate with other nodes to obtain a variety of information contained in log files of other nodes in the recovery process of the crashed node. In sharp contrast, Claim 10 requires that if a failure involves only a first node, recovery is performed by applying a recovery log of the first node beginning at a first point in the recovery log, and if the failure involves one or more nodes in addition to the first node, then recovery is performed by applying the recovery log of the first node beginning at a different point in the same recovery log of the crashed node. This is not shown by *Biliris*.

Instead, the portions of *Biliris* cited by the Office Action cited the show the elements of Claim 10 (Col. 9; lines 35-50; Col. 10, lines 8-12; Col. 10, lines 66-67; Col. 11; lines 2-14; Col. 12, lines 60-67; Col. 13, lines 3-12) merely describes a system where during the recovery of a crashed node, the crashed node has to (a) determine the pages that may require recovery, (b) identify the nodes involved in the recovery, (c) reconstruct lock information, and (d) coordinate the recovery among the involved nodes (see Col. 9, lines 46-49).

Thus, regardless of whether the failure involves a single node or two or more nodes, in the approach of *Biliris*, recovery is not performed by applying a recovery log of the node that failed. Instead, the cited portions of *Biliris* describe performing recovery of a crashed node by the crashed node engaging in a series of communications between other nodes to obtain information about which pages need to be recovered, which nodes are involved in the recovery, reconstructing lock information, and coordinating the recovery among the involved nodes. These communications involve scanning the log files of other nodes besides the crashed node.

To illustrate, *Biliris* teaches when a crashed node restarts, "it requests from each operational node N, the list of all pages owned by N that are present in a remote node's (N_r's) cache, as well as all entries in N_r's DPT that correspond to pages owned by N" (See Col. 10; lines 41-44). Further, *Biliris* teaches "when a remote node N_r receives the list of pages that require recovery from N, it scans its log file starting from the minimum of all RedoLSN values belonging to DPT entries for the pages that are included in the above list (see Col. 11, lines 57-60). Thus, it is clear that in the approach of *Biliris*, the recovery of a failed node cannot be performed by applying a recovery log of the failed node, as featured in Claim 10.

As a result, *Biliris* cannot disclose, teach, or suggest the elements of "if the failure involves only said one node, then performing recovery by applying a recovery log of said node beginning at a first point in the recovery log; and if the failure involves one or more nodes in addition to said

one node, then performing recovery by applying said recovery log of said node beginning at a second point in the recovery log; wherein said first point is different from said second point" featured in Claim 10.

As *Biliris* fails to disclose, teach, or suggest one or more elements recited in Claim 10, it is respectfully submitted that Claim 10 is patentable over the cited art, and is in condition for allowance.

Claim 12

Claim 12 recites the features of:

> A method for recovering after a failure, the method comprising the steps of:
> if it is unclear whether a particular version of a data item has been written to disk, then performing the steps of
> without attempting to recover said data item, marking dirtied cached versions of said data item that would have been covered if said particular version was written to disk;
> when a request is made to write one of said dirtied cached versions to disk, determining which version of said data item is already on disk; and
> if said particular version of said data item is already on disk, then not writing said one of said dirtied cached versions to disk.

The above combination of elements is not disclosed, taught, or suggested by *Peter*.

Peter is directed towards recovery in large-scale transaction-based distributed systems with replicated data. *Peter* teaches that distributed systems require the execution of a two-phase commit protocol. In the first phase (the "prepare phase"), the coordinator queries and gathers votes form the participant processes as to whether they are willing to commit. If any process refuses to commit the transaction, the coordinator sends in the second phase an "abort" message to all participants. Otherwise, the coordinator sends a "commit" message to all participant processes to instruct them to commit the transaction.

The Office Action cites *Peter* (page 40, Col. 2) to show the element of "without attempting to recover said data item, marking dirtied cached versions of said data item that would have been covered if said particular version was written to disk" featured in Claim 12. However, this portion lacks any suggestion of marking any dirty cached versions of any data item, let alone a data item that would have been covered if a particular version of the data item, which may or may not have been written to disk, was actually written to disk.

Instead, this portion of *Peter* discusses recovering from a crash when a node contains "uncertain transactions in the log." An uncertain transaction, according to *Peter*, is a transaction that has a "prepared" entry, but not corresponding "commit" or "abort" entry in the log. Thus, the techniques discussed in this section of *Peter* relate to how to recover from a crash when a coordinator may or may not have issued a commit instruction for a particular transaction. However, this section does not discuss, or suggest, how to process data items stored in a cache. Thus, this portion of *Peter* cannot disclose, teach, or suggest this element.

Peter is also cited (at page 40, Col. 2, items 1 and 2) to show the element of "when a request is made to write one of said dirtied cached versions to disk, determining which version of said data item is already on disk" featured in Claim 12. However, this portion of *Peter* lacks any discussion of (a) a cache, (b) a dirty version of a data item stored in a cache, (c) receiving a request to write anything stored in cache to disk, and (d) determining which version of a data item is stored in disk. Consequently, this portion of *Peter* cannot possibly disclose, teach, or suggest this element.

Peter is further cited (at page 40, Col.2, items 1 and 2) to show the element of "if said particular version of said data item is already on disk, then not writing said one of said dirtied cached versions to disk" featured in Claim 12. However, this portion of *Peter* lacks any discussion of (a) determining whether a particular version of a data item is already on disk, and (b)

writing a dirty version of a data item from cache to disk. Consequently, this portion of *Peter* cannot possibly disclose, teach, or suggest this element.

As *Peter* fails to disclose, teach, or suggest one or more elements recited in Claim 12, it is respectfully submitted that Claim 12 is patentable over the cited art, and is in condition for allowance.

Claims 6-9, 11, 13-14, and 23-32

Independent Claim 23 recites features similar to those discussed above with respect to Claim 5, except that Claim 23 is recited in computer-readable medium format. Independent Claim 28 recites features similar to those discussed above with respect to Claim 10, except that Claim 28 is recited in computer-readable medium format. Independent Claim 30 recites features similar to those discussed above with respect to Claim 12, except that Claim 30 is recited in computer-readable medium format. Consequently, it is respectfully submitted that Claims 23, 28, and 30 are patentable over the cited art, and are each in condition for allowance, for at least the reasons given above with respect to Claims 5, 10, and 12 respectively.

Claims 6-9, 11, 13-14, 24-27, 29, and 31-32 are dependent claims, each of which depends (directly or indirectly) on one of the claims discussed above. Each of Claims 6-9, 11, 13-14, 24-27, 29, and 31-32 is therefore allowable for the reasons given above for the claim on which it depends. In addition, each of Claims 6-9, 11, 13-14, 24-27, 29, and 31-32 introduces one or more additional limitations that independently render it patentable. However, due to the fundamental differences already identified, to expedite the positive resolution of this case, a separate discussion of those limitations is not included at this time. The Applicants reserve the right to further point out the differences between the cited art and the novel features recited in the dependent claims.

CONCLUSION

For the reasons set forth above, it is respectfully submitted that all of the pending claims are now in condition for allowance. Therefore, the issuance of a formal Notice of Allowance is believed next in order, and that action is most earnestly solicited.

The Examiner is respectfully requested to contact the undersigned by telephone if it is believed that such contact would further the examination of the present application.

A petition for extension of time, to the extent necessary to make this reply timely filed, is hereby made. If any applicable fee is missing or insufficient, throughout the pendency of this application, the Commissioner is hereby authorized to charge any applicable fees and to credit any overpayments to our Deposit Account No. 50-1302.

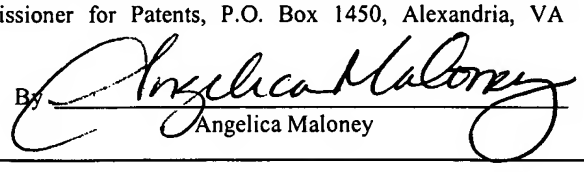
Respectfully submitted,
HICKMAN PALERMO TRUONG & BECKER LLP

Christopher J. Brokaw
Reg. No. 45,620
Dated: June 9, 2005

2055 Gateway Place, Suite 550
San Jose, California 95110-1089
Telephone: (408) 414-1080 ext. 225
Facsimile: (408) 414-1076

CERTIFICATE OF MAILING

I hereby certify that this correspondence is being deposited with the United States Postal Service as first class mail in an envelope addressed to: **Mail Stop Amendment**, Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450.

On June 9, 2005 By 
Angelica Maloney